

## CLAIMS

What is claimed is:

1. A method for creating finite state automata (FSA) that match patterns in parallel, comprising:  
creating states of the finite state automata from a set of patterns to be matched;  
passing over the set of patterns a second time; and  
adding transitions to the states to match all possible patterns that can start within the set of patterns to be matched.
2. The method of claim 1 further comprising:  
iterating through the states;  
determining whether input causes a move to an initial state; and  
if the initial state has a different move on the input, changing a current state's transition to mirror that of the initial state.
3. A method of creating a FSA that uses array-based transitions for an alphabet of size N, comprising:  
representing each state as an object containing an array of N pointers to possible successive states;  
using a numeric value of each member of the alphabet as an offset into the array to point to a next state.
4. A method of creating a case-insensitive FSA by making each pattern all one case, comprising:  
creating the FSA; and  
adding corresponding transitions on each alphabetic character so that case of characters in an input stream will have no effect on performance of the case-insensitive FSA.

5. A method for matching patterns, comprising:  
using a numeric value of less than a complete set of bits of an input as an offset into  
an array, thereby reducing a size of the array.
6. The method of claim 5 comprising a further step of using a hash function for  
matching patterns composed of a 128 or 256 alphabet without overhead of larger  
arrays.